Featherweight Breeze: Step 3/4

Cătălin Hriţcu, Benoît Montagu, Benjamin C. Pierce, and the Breeze team

December 8, 2011

1 Syntax

$L,\ H,\ pc$	::=			label
	|	$\top$	M	top secret
	|	$\bot$	M	unclassified
	|	$L_1 \vee L_2$	M	label join
	|	(L)	S	
c	::=			constants
	|	()		unit
	|	true		true
	|	false		false
	|	L		label
t	::=			terms
	|	c		constant
	|	x		variable
	|	$\lambda x.t$	bind x in t	abstraction
	|	$t_1\ t_2$		application
	|	let $x = t_1$ in t_2	bind x in t_2	let
	|	(t_1, t_2)		pairing
	|	fst t		first projection
	|	snd t		second projection
	|	if t_1 then t_2 else t_3		conditional
	|	$t_1 == t_2$		equality on constants
	|	$t_1 @ t_2$		classify t_1 with label t_2
	|	$t_1\langle t_2\rangle$		executes t_2, labels result with t_1, restores pc
	|	labelOf t		returns the label of t
	|	getPc ()		returns the current pc
	|	valueOf t		takes label of t and joins it to pc

	$[t]$	M	wrong attempt to define brackets uses this
	(t)	S	

v	::=			values
	|	c		constants
	|	$\langle \rho,\, \lambda x.\, t \rangle$	bind x in t	closures
	|	(a_1, a_2)		pairs

a	::=			atoms
	|	$v@L$		labeled value

ρ	::=			environments
	|	*empty*		
	|	$\rho, x : a$		
	|	(ρ)	S	

2 Evaluation with Dynamic IF Control

$\boxed{\rho \vdash t, pc \Downarrow a, pc'}$

$$\frac{}{\rho \vdash c, pc \Downarrow c@\bot, pc} \quad \textsc{Eval\_Const}$$

$$\frac{\rho(x) = a}{\rho \vdash x, pc \Downarrow a, pc} \quad \textsc{Eval\_Var}$$

$$\frac{}{\rho \vdash (\lambda x.t), pc \Downarrow \langle \rho,\, \lambda x.\, t \rangle @ \bot, pc} \quad \textsc{Eval\_Abs}$$

$$\frac{\begin{array}{l}\rho \vdash t', pc \Downarrow \langle \rho',\, \lambda x.\, t \rangle @ L', pc' \\ \rho \vdash t'', pc' \Downarrow a'', pc'' \\ (\rho', x : a'') \vdash t, (pc'' \vee L') \Downarrow a, pc'''\end{array}}{\rho \vdash (t'\ t''), pc \Downarrow a, pc'''} \quad \textsc{Eval\_App}$$

$$\frac{\begin{array}{c}\rho \vdash t', pc \Downarrow a', pc' \\ (\rho, x : a') \vdash t'', pc' \Downarrow a'', pc''\end{array}}{\rho \vdash (\mathsf{let}\ x = t'\ \mathsf{in}\ t''), pc \Downarrow a'', pc''} \quad \textsc{Eval\_Let}$$

$$\frac{\begin{array}{c}\rho \vdash t', pc \Downarrow a', pc' \\ \rho \vdash t'', pc' \Downarrow a'', pc''\end{array}}{\rho \vdash (t', t''), pc \Downarrow (a', a'') @ \bot, pc''} \quad \textsc{Eval\_Pair}$$

$$\frac{\rho \vdash t, pc \Downarrow (v'@L', a'')@L, pc'}{\rho \vdash (\mathsf{fst}\ t), pc \Downarrow v'@L', (pc' \vee L)} \quad \textsc{Eval\_Fst}$$

$$\frac{\rho \vdash t, pc \Downarrow (a', v''@L'')@L, pc'}{\rho \vdash (\mathsf{snd}\ t), pc \Downarrow v''@L'', (pc' \vee L)} \quad \textsc{Eval\_Snd}$$

$$\frac{\rho \vdash t, pc \Downarrow \mathsf{true}@L, pc' \quad \rho \vdash t', (pc' \vee L) \Downarrow a', pc''}{\rho \vdash (\mathsf{if}\ t\ \mathsf{then}\ t'\ \mathsf{else}\ t''), pc \Downarrow a', pc''} \quad \textsc{Eval\_If\_True}$$

$$\frac{\rho \vdash t, pc \Downarrow \mathsf{false}@L, pc' \quad \rho \vdash t'', (pc' \vee L) \Downarrow a'', pc''}{\rho \vdash (\mathsf{if}\ t\ \mathsf{then}\ t'\ \mathsf{else}\ t''), pc \Downarrow a'', pc''} \quad \textsc{Eval\_If\_False}$$

$$\frac{\rho \vdash t', pc \Downarrow c'@L', pc' \quad \rho \vdash t'', pc' \Downarrow c''@L'', pc'' \quad v \triangleq c' = c''}{\rho \vdash (t' == t''), pc \Downarrow v@(L' \vee L''), pc''} \quad \textsc{Eval\_Eq}$$

$$\frac{\rho \vdash t, pc \Downarrow v@L, pc' \quad \rho \vdash t', pc' \Downarrow L'@L'', pc''}{\rho \vdash (t@t'), pc \Downarrow v@(L \vee L'), (pc'' \vee L'')} \quad \textsc{Eval\_Classify}$$

$$\frac{\rho \vdash t', pc \Downarrow L@L', pc' \quad \rho \vdash t'', (pc' \vee L') \Downarrow v@L'', pc'' \quad L'' \vee pc'' \sqsubseteq L \vee (pc' \vee L')}{\rho \vdash t'\langle t''\rangle, pc \Downarrow v@L, (pc' \vee L')} \quad \textsc{Eval\_Bracket}$$

$$\frac{\rho \vdash t, pc \Downarrow v@L, pc'}{\rho \vdash \mathsf{labelOf}\ t, pc \Downarrow L@\bot, pc'} \quad \textsc{Eval\_LabelOf}$$

$$\frac{}{\rho \vdash \mathsf{getPc}\ (), pc \Downarrow pc@\bot, pc} \quad \textsc{Eval\_GetPc}$$

$$\frac{\rho \vdash t, pc \Downarrow v@L, pc'}{\rho \vdash \mathsf{valueOf}\ t, pc \Downarrow v@\bot, (pc' \vee L)} \quad \textsc{Eval\_ValueOf}$$

3 Brackets

Brackets are constructs for executing a computation and restoring the initial pc when the computation ends.

3.1 First try

$$\frac{\rho \vdash t, pc \Downarrow v@L, pc'}{\rho \vdash [t], pc \Downarrow v@(L \vee pc'), pc}$$

The main idea is to move some of the protection from the pc to the label on the resulting value. Since v is protected in the premise by L and by pc', in the result we can move all this protection to the label of v, which is now $L \vee pc'$, and the pc can safely be restored to the original one. The reason this doesn't quite work is that labels are public, and while in the premise the label L is protected by pc', in the conclusion L would only be protected by the (potentially lower) pc. Here is a counterexample exploiting the label channel:
$\mathsf{let}\ y = [\mathsf{if}\ x@H\ \mathsf{then}\ ()@H\ \mathsf{else}\ ()@\top]\ \mathsf{in}\ \mathit{publish}\ (\mathsf{labelOf}\ y) == H$

Another problem is that in the premise pc' is protected by itself, while in the conclusion pc' is protected only by pc. The counterexample for this looks as follows:
$\mathsf{let}\ y = [\mathsf{if}\ x@H\ \mathsf{then}\ \mathit{raisePc}\ H\ \mathsf{else}\ \mathit{raisePc}\ \top]\ \mathsf{in}\ \mathit{publish}\ (\mathsf{labelOf}\ y) == H$
where $\mathit{raisePc} \triangleq \lambda x.((\lambda y.y)@x)\ ()$ (in step 4/4 we'll add a *raisePc* primitive).

3.2 Second try: closing the label channel

$$\frac{\rho \vdash t, pc \Downarrow v@L'', pc'' \quad L'' \vee pc'' \sqsubseteq L}{\rho \vdash L\langle t\rangle, pc \Downarrow v@L, pc}$$

We close the label channel by requiring the user to choose in advance the label on the result. This way the label on the result cannot depend on secrets. This works but is still too restrictive: because of the $L'' \vee pc' \sqsubseteq L$ condition in the premise we cannot use brackets to classify values to a low label in a high context. For instance $\mathsf{if}\ x@\top\ \mathsf{then}\ \bot\langle\mathsf{true}\rangle\ \mathsf{else}\ ()$ fails, although $\mathsf{if}\ x@\top\ \mathsf{then}\ \mathsf{true}@\bot\ \mathsf{else}\ ()$ works fine.

3.3 Third try: making brackets the ultimate classification construct

$$\frac{\rho \vdash t, pc \Downarrow v@L'', pc'' \quad L'' \vee pc'' \sqsubseteq L \vee pc}{\rho \vdash L\langle t\rangle, pc \Downarrow v@L, pc}$$

The intuition is that the final value is not only protected by L, but also by the pc, so we can relax the premise of the rule from $L'' \vee pc' \sqsubseteq L$ to $L'' \vee pc' \sqsubseteq L \vee pc$. This works but is still too restrictive, since the label L is required to be a constant.

3.4 Fourth try: first-class labels on brackets

$$\frac{\rho \vdash t', pc \Downarrow L@\bot, pc' \quad \rho \vdash t'', pc' \Downarrow v@L'', pc'' \quad L'' \vee pc'' \sqsubseteq L \vee pc}{\rho \vdash t'\langle t''\rangle, pc \Downarrow v@L, pc}$$

This step is very easy, but only as long as the label on L is required to be $\bot$.

3.5 Fifth try: the final rule

The final Eval_Bracket rule additionally takes care of the label on L by raising the pc appropriately, otherwise it's the same as before.

4 Other Changes wrt Step 2

- Droped automatic pc lowering/restoring. Now threading the pc through as a piece of state.
- Made pc be non-infectious.
- Made all labels public: labelOf no longer protects the resulting label with itself, and added a new getPc construct.

- The old rules for pair projection (EVAL_FST and EVAL_SND) are unsound in our new public label setting. The fix is to join the outer pair label to the resulting *pc*.
- The old rule for classification would also be unsound in our new public label setting; fixed. Classification is anyway completely subsumed by brackets: $t_1 @ t_2 \triangleq \mathsf{let}\, x = t_2\, \mathsf{in}\, x\langle t_1 \rangle$.
- Added new valueOf construct that strips off the label of an atom and joins it to the *pc*. This is roughly the dual of brackets, which take taint from the *pc* and put it in the label of the result.

5 Counterexamples

- Here is why EVAL_FST and EVAL_SND had to change:
  $\mathsf{let}\, y = H\langle \mathsf{if}\, x @ H\, \mathsf{then}\, ((), H), ()) \,\mathsf{else}\, ((), \top), ())\rangle\, \mathsf{in}\, \mathit{publish}\, (\mathsf{labelOf}\, (\mathsf{fst}\, y) == H)$

6 This Fixes the Two Problems from Step 2

6.1 The "Infectious *pc*" Problem Fixed

$\mathit{empty} \vdash H\langle \mathsf{if}\, (\mathsf{true} @ H)\, \mathsf{then}\, (\mathsf{true}, (\mathsf{false}, ()))\, \mathsf{else}\, ()\rangle, \bot \Downarrow (\mathsf{true} @ \bot, (\mathsf{false} @ \bot, () @ \bot) @ \bot) @ H, \bot$

6.2 The "Poison Pill" Problem Fixed

- Labels are now public.
- Critical components can use labelOf to protect themselves from "poison pills".
- IFC violations no longer need to be fatal errors (still a lot of care needs to be used when adding exceptions, they don't interact too well with brackets).